AFRL-IF-WP-TM-2002-1567

THE CARNEGIE MELLON UNIVERSITY INSERT PROJECT

John Lehoczky
Lui Sha
Bruce Krogh
Peter Feiler
Ragunathan Rajkumar

Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213-3890

FEBRUARY 1997

Final Report for 01 October 1996 – 28 February 1997

INFORMATION DIRECTORATE
AIR FORCE RESEARCH LABORATORY
AIR FORCE MATERIEL COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OH 45433-7334

[illegible]
KENNETH LITTLEJOHN
Project Engineer

[illegible]
JAMES S. WILLIAMSON, Chief
Embedded Info Sys Engineering Branch
Information Technology Division
Information Directorate

[illegible]
WALTER B. HARTMAN
Acting Wright Site Coordinator
Information Directorate

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. REPORT DATE ***(DD-MM-YY)*** February 1997	**2. REPORT TYPE** Final	**3. DATES COVERED** ***(From - To)*** 10/01/1996 – 02/28/1997

4. TITLE AND SUBTITLE THE CARNEGIE MELLON UNIVERSITY INSERT PROJECT	**5a. CONTRACT NUMBER** F33615-96-2-1948
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER 62301E
6. AUTHOR(S) John Lehoczky, Lui Sha, Bruce Krogh, Peter Feiler, and Ragunathan Rajkumar	**5d. PROJECT NUMBER** ARPA
	5e. TASK NUMBER AA
	5f. WORK UNIT NUMBER 18
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213-3890 / Defense Advanced Research Projects Agency, Information Technology Office, 3701 North Fairfax Drive, Arlington, VA 22209-2308	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Information Directorate, Air Force Research Laboratory, Air Force Materiel Command, Wright-Patterson Air Force Base, OH 45433-7334	**10. SPONSORING/MONITORING AGENCY ACRONYM(S)** AFRL/IFTA
	11. SPONSORING/MONITORING AGENCY REPORT NUMBER(S) AFRL-IF-WP-TM-2002-1567

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT ***(Maximum 200 Words)***
This document constitutes the final report to the revised Statement of Work for the Carnegie Mellon University Incremental Software Evolution for Real-Time Systems (INSERT) project under the DARPA Evolutionary Design for Complex Software (EDCS) Program.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT:**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** (Monitor)
a. REPORT Unclassified	**b. ABSTRACT** Unclassified	**c. THIS PAGE** Unclassified	SAR	22	Kenneth Littlejohn **19b. TELEPHONE NUMBER** ***(Include Area Code)*** (937) 255-6548 x3587

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18

FINAL REPORT
The Carnegie Mellon University INSERT Project
Contract Number F33615-96-2-1948
October 1, 1996 to February 28, 1997

John P. Lehoczky, PI: jpl@stat.cmu.edu
Lui Sha: lrs@sei.cmu.edu
Bruce Krogh: krogh@ece.cmu.edu
Peter Feiler: phf@sei.cmu.edu
Ragunathan Rajkumar: raj@sei.cmu.edu

Abstract

This document constitutes the final report to the revised Statement of Work for the Carnegie Mellon University INSERT Project under the DARPA EDCS Program as described in Appendix 1. The period covered is October 1, 1996 to February 28, 1997.

Task 1: Application Independent Run-Time Capabilities

During this five month period, CMU will begin work on Subtasks 1.1, 1.2 and 1.4.

Subtask 1.1

CMU will begin research to develop the fault tolerant publisher/subscriber inter-process communications abstraction.

Objective

Providing flexible real-time fault tolerant group communication mechanisms is an important INSERT task. The goal of this task is to extend the existing real-time publisher/subscriber mechanism in the existing Simplex architecture to ensure that the states of the distributed components of the real-time publisher/subscriber will remain consistent even in the event of a crash or other failures of nodes.

Summary of current progress to date

The INSERT team has completed an initial API definition and ported the existing real-time publication subscription group communication software to LynxOS 2.4, a POSIX.1b compliant OS. The distributed real-time publisher/subscriber communication model is now supported by a processor membership protocol which allows a node in the system to fail, or to rejoin the system later. When a node fails, all the publishers and subscribers on that node have to be deleted from the publisher/subscriber information maintained by the middleware layer. When a node tries to rejoin, the current state information about the publishers/subscribers must be transferred to the node before it can integrate into the system. A two-phase protocol is used to ensure that the state information from the requested node is transferred in consistent fashion (since the information can be changing when the transfer happens). The message distribution list for distributed group communication is currently undergoing further evaluation in the context of real-time control experiments. Other areas to be completed in the publisher/subscriber model include:

- Information query support: Interrogation of message distribution lists information.
- Checkpointing: Distributed checkpointing and recovery for message distribution list.

- Statistics collection: Detailed error progress to date about information for multicast sends.

Finally, we are making significant enhancements to the publisher/subscriber communication model to address the following lack of flexibility in the publisher/subscriber communication model.

- Information consumers need not be running and consuming information at the same rate as the information producers, and
- The publisher/subscriber communication model is synchronous: a subscriber must be listening in order to receive messages by a publisher.

Both these constraints will be relaxed by a real-time push-pull communication model, which will be a powerful generalization of the real-time publisher/subscriber communication model. The real-time push/pull communication model enables new components with higher *or* lower capabilities to be inserted into a current system without requiring the existing information publishers to change. This yields significant benefits in the design, development and deployment stage of an evolving long-lived system. Crash-failures of nodes can be detected and recovered from. This yields robustness and satisfaction of critical functionality at run-time (since the capability of the system evolves at run-time due to node failure or due to node recovery).

A detailed design of the real-time push=pull model and an application programming interface have been developed. The processor membership protocol, which allows node failures to be tolerated, is a largely orthogonal aspect enabling quick integration into the future. A description of this general model is presented in the appendix.

Subtask 1.2

Objective:
CMU will develop the initial design of a *distributed clock synchronization scheme* for use in the INSERT middleware. The finalization and implementation of the design will be done under the follow-on contract.

Summary of current progress to date:
A distributed clock synchronization scheme based on Flaviu Cristian's algorithm has been implemented in LynxOS. Cristian's algorithm is probabilistic if the communication delays between nodes do not have bounded (or deterministic) delays. Since the INSERT testbed is generally closed with only controlled tasks in the system, we assign an appropriately high priority to the clock synchronization daemon on each node. We therefore obtain not only deterministic delays (bounded worst-case delays) but also very low delays by the use of a high-bandwidth switched network.

Subtask 1.4

Objective
CMU will initiate collaborations with other EDCS contractors in the High Assurance Cluster and in the Run-time Correction Preservation Affinity Group.

Summary of current progress to date
The EDCS program has been divided into five distinct project clusters, and the CMU INSERT project is participating in the **High Assurance and Real-Time Cluster**. Cluster participants met twice during the contract period with a third meeting schedule for April 1-2, 1997.

The first Cluster meeting was held in July 1996 at the initial annual EDCS meeting. At that meeting, the Cluster participants as well as many investigators involved in formal methods research described their individual research projects. Then, cluster participants worked to develop a broad scenario within which all project could contribute. This scenario involved aspects of the entire lifecycle of system evolution. The INSERT Project was especially concerned with three aspects: (1) the development of technology to support on-line upgrades along with the development of a safety net which would protect the system should the upgraded modules fail, (2) dependency tracking technology to determine all the changes that must be made when one part of the system is changed and (3) factorial testing methodology to permit an entire test sequence to be performed, even if a failure occurs in any part of the sequence.

Further developments occurred at the second cluster meeting held on October 9-10, 1996 in Dallas. At this meeting, the INSERT F-16 vignette was proposed for adoption as the cluster vignette against which all projects could work. The cluster meeting also presented an opportunity for discussions among various project personnel to find avenues for cooperation. Preliminary discussions with Honeywell took place, and the possibility of INSERT using Honeywell's Meta-H and Control-H for dependency tracking was explored. Separately from the Cluster meetings, INSERT personnel continue to cooperate closely with Carol Hoover of the CMU METAPHOR project.

In addition to the cluster meetings, the INSERT project is an Integrating project in the cluster, that is it will provide a vehicle for certain selected EDCS projects to demonstrate their technology. Two Project Integrator Meetings were held during the contract period, on in December 1996 and a second in February 1997. As a consequence of all this cluster and integrator activity, it now appears that CMU will either collaborate with or integrate technology from the Honeywell Meta-H project (for dependency tracking), the CMU METAPHOR project (for change analysis and composable real-time software), David Garlan and Daniel Jackson of CMU for formally describing INSERT using architectural descriptions languages and Bob Balzer of USC-ISI on the use of instrumented connectors. Discussions with Lee Osterweil, Lori Clark, Debra Richardson and Michael Young on integrating various aspects of testing methodology are on-going.

Task 2: Avionics Domain-Specific Components

During this contract period, CMU will begin work on Subtasks 2.1 and 2.4.

Subtask 2.1

Development of a formal model for the INSERT runtime switching logic.

Objective

INSERT supports the reliable online upgrade of controllers. The switching logic is a rule for switching the controllers based on the behavior of the physical system to be controlled. It is the key element to guarantee the safe upgrade by evaluating two aspects of the new controller: 1) Does the physical system under the new controller perform within the domain in which the safety controller can handle in case a fault occurs? and 2) Does the new controller result in a satisfactory performance of the physical system? The current focus of this work is to develop systematic approaches to derive control switching logic. This amounts to abstracting the runtime continuous dynamics of the physical system into discrete descriptions. There is no standard methodology that one can follow to work out such abstraction, but different areas in dynamics and control theory need to be investigated.

Summary of progress to date

We have been focusing on systematically developing safety switching rules, i.e., the logic which leads to control switching from the new controller to the safety controller when a fault due to the upgrade is detected. Specifically, we formally define the notion of safety region and safety control, the safety of the physical system, and controller's operational region. The development of control switching logic or the abstraction of the continuous dynamics in terms of safety then allow one to identify the safety region of the physical system and to design the safety controller to maintain the state of the physical system inside the safety region. Towards this end, three approaches were examined.

Design the safety controller as time-optimal

Control Goal: Driving the physical system to a prescribed region in the minimum time and subject to state constraints.

Safety Criterion: If the control goal can be reached from a given state, that physical system is safe at that state. Otherwise it is unsafe.

Safety check implementation: the nonlinearity in the physical systems makes the evaluation of the safety criterion difficult to be carried out in practice. There are two approximations can be applied to address this problem

- Conservative linearization: we (piecewisely) linearize the physical system in such a way that if the linearized system satisfy the safety criteria, so will the nonlinear system.
- Real-time simulation: when the simulation computation is faster than the dynamics of the physical system, it is possible to use simulation to determine if the control goal can be achieved.

Design the safety controller as stabilization control

Control Goal: Stabilize the physical system at an equilibrium state using state feedback control and subject it to state and control constraints. This is achieved in three steps. First, design a stabilization controller with a Lyapunov function which describes the domain of attraction (DOA). Second, identify the intersection of the set of admissible states derived from the limits of the control and the set of admissible states obtained from the state constraints. Finally, adjust the parameter of the Lyapunov function such that the DOA just fits inside the intersection.

Safety Criterion: The physical system is safe if its state is inside the DOA. Otherwise it is unsafe.

Neural-Network-Based Switching Rules

To generate run-time switching rules automatically, we are investigating the application of neural networks which can "learn" the regions of stability and the performance indices for different controllers. Simulation examples have demonstrated the viability of this approach, and hardware experiments are being implemented. Two conference papers have been written describing this research direction.

While the first approach does not result in an explicit safety region, it generates the largest set of safe states of the physical system. On the other hand, the second approach gives a closed-form analytical expression of the safety region, but such region may be somewhat conservative. We have successfully implemented the algorithm developed from both approaches in a navy control application, i.e., a diver control system, and we expect to apply them to avionic control systems to establish the INSERT run-time switching logic.

Subtask 2.4

CMU will investigate systematic methods for constructing reduced-order models of complex nonlinear dynamics.

Objective

The INSERT architecture for real-time control applications leads to the complex interaction of switching control logic with continuous control loops. The dynamics are nonlinear in many respects, not simply because of possible nonlinearities in the process being controlled. There are currently no general methods for analyzing the properties of such systems, yet it is essential to be able to analyze and verify the properties of INSERT-based control implementations. There is also no existing methodology for defining the run-time switching rules. Methods need to be developed to create reduced-order models of the complex nonlinear dynamics so that switching rules can be defined and analysis can be performed for applications of the INSERT technology.

Summary of current progress to date

Hybrid System Modeling Tools

The interaction of switching with continuous control loops creates so-called hybrid systems. Only recently have computer modeling tools began to emerge that provide adequate representations of both continuous and discrete dynamics in an integrated package. We have initiated work using the MATLAB/SIMULINK tool that has a new finite-state machine capability to build complex models of the dynamics that arise in INSERT applications. We have constructed and simulated examples of the run-time switching rules for the inverted pendulum example to become familiar with this tool.

We also investigated the use of qualitative differential equations as a method for modeling processes being controlled with the INSERT run-time system. QSIM, a tool from UT Austin, performs qualitative simulations and analysis of such models. From our investigations and experiments, it appears that the analysis one can perform with this tool is much too conservative to be of value in serious applications. Some of the fundamental concepts may be useful, however, in developing more relevant reduced-order models of the nonlinear dynamics in INSERT applications.

As a fundamental approach to reduced-order model analysis, we are advocating the development of finite-state models of the continuous dynamics which can be coupled to and analyzed with the switching logic in an INSERT run-time environment. Towards that end, we have begun exploring techniques for generating finite-state models directly from the continuous state equations. Two techniques currently under investigation are: (i) the generation of outer approximations for the continuous dynamic flows; and (ii) a partial differential equation technique for propagating switching thresholds in the continuous domain.

Neural-Network-Based Switching Rules

To generate run-time switching rules automatically, we are investigating the application of neural networks which can "learn" the regions of stability and the performance indices for different controllers. Simulation examples have demonstrated the viability of this approach, and hardware experiments are being implemented. Two conference papers have been written describing this research direction.

Possible methods to simplify the model of the complex nonlinear systems:

1. Coordinate transformation to decouple the complex nonlinear dynamics.

2. Introduce state feedback to linearize the decoupled nonlinear systems.

3. Linearize the nonlinear dynamics at an equilibrium state.

Methods 1 and 2 have been successfully applied in studying complex aircraft dynamics, and we expect to use them in avionic control systems as well. Method 3 is the standard approach to deal with nonlinear systems, and we may find some use for it in case Methods 1 and 2 do not work.

Task 3: Dependency Analysis Methods and Tools

Subtask 3.1
CMU will investigate current methodologies available for design dependency tracking. CMU will also contact other EDCS contractors to investigate potential collaborations on this topic and begin the assessment of tools which are already available or will become available within an 18 month period.

Objective
The INSERT architecture allows components to be replaced incrementally and online. Furthermore, in case of failure of such a component the safety-net switching rules will safely revert to the baseline component. Such changes have impact on other parts of the system in terms of timing, resource, and semantic dependencies. The purpose of this subtask is to assess these methods and tools for their capability and limitation to address the stated problem, and for their consideration as prototyping platform to overcome the limitations.

Summary of current progress to date
Our approach is to determine an initial set of modeling requirements, and to assess existing methods and tools in light of these requirements.

First, we examined the modeling requirements. Schedulability and resource dependency issues are addressed at the granularity of processes, communication connections, and shared objects, i.e., at the level of implementation architectures. Semantic dependencies require some aspects of the application domain to be captured that facilitate impact analysis in terms of the feedback control system domain as well. Two major types of controllers can be identified: continuous controllers, and supervisory controllers. Continuous controllers are based on a set of control equations. Supervisory controllers model discrete states and events and are used to represent operation sequencing and operational modes. INSERT technology extends the concept of mode, i.e., the ability to dynamically switching between predetermined sets of operational capabilities, by supporting runtime replacement of components. Modeling notations need to accommodate analysis of consistent system configuration variants. In summary, the modeling capabilities need to be able to capture semantic information associated with components of control systems, to identify consistent combinations of component variants, and relate them to the implementation domain in terms of processes and communication connections in the realm of hard real-time applications.

We proceeded with a hands-on evaluation of two candidates of Architectural description languages (ADL): Rapide by Stanford University, and Meta-H by Honeywell. The evaluation was accomplished by modeling INSERT specific concepts and by investigating necessary extensions to accommodate these concepts. Rapide represents a general architectural description language with emphasis on structural modeling, and event-based behavioral modeling based on partially ordered sets. Its focus is on modeling and analysis of software system architectures. In contrast, Meta-H and its tool support focuses on supporting the development of hard real-time systems, in particular in the domain of guidance and control. Meta-H models software system architectures in terms of processes with real-time characteristics and their interaction in terms of communication, events, and alternative configurations in terms of interconnected sets of processes. Its toolset support both analysis and synthesis. The analysis capability in Meta-H includes validation of interface properties as well as schedulability analysis, including sensitivity analysis (i.e., it takes into consideration possible variation in execution time), of set of processes. Their mapping onto particular hardware architectures is also specified in Meta-H. The synthesis component is able to generate an runtime infrastructure that correctly implements the specified scheduling constraints and communication structures.

We modeled the uniprocessor implementation of the inverted pendulum. Due to the nature of Rapide the resulting system model focused on the interaction protocol of several components of the runtime system and the INSERT switching logic. Schedulability aspects could not be addressed. Comprehending the power of the full language and applying it successfully has a high learning curve. The toolset takes a traditional language compiler approach, i.e., a textual representation of a system expressed in Rapide is processed for analysis by invoking the tool on a file. Extensions to the toolset would have to be negotiated with the Rapide development team.

We then proceeded to examine the Honeywell toolset, consisting of Meta-H and Control-H as modeling languages, a Meta-H compiler, and Control-H compiler, and a graphical interactive editor front end to both Meta-H and Control-H created through Meta-DOME. Meta-DOME is an interactive graphical editor prototyping facility in an object-oriented environment, that allows instances of editors for various languages to be developed interactively and for those editors to perform various forms of analysis, generation of textual representations, and interfacing with other tools. Various parts of the toolset have been in use by the Honeywell Technology, other parts of Honeywell, as well as outside parties.

We modeled the process and communication pattern present in a set of analytically redundant controllers in INSERT. In the process we were able to accomplish two things. First, we were able to identify an application level abstraction for INSERT-based development referred to as Analytically Redundant Component (ARC), which embodies the details of how to implement a collection of analytically redundant controller variants and guarantee dependable operation. Second, we were able to determine that Meta-H provides a good set of modeling concepts, but lacks the ability to specify a delay in process start time, resulting in a conservative schedulability result. We have identified work by Tindell at York University to overcome this shortcoming. We have demonstrated that their algorithm can perform a more optimistic schedulability analysis. We have also shown that the sensitivity analysis can be extended to the Tindell algorithm.

An initial examination of control system design languages revealed that notations for modeling continuous control provide little explicit support for structuring. Honeywell's toolset - integrating Control-H and Meta-H - is a good illustration of how such a shortcoming can be compensated for by combining such control engineering tools with software system engineering tools. Modeling notations for supervisory control are typically based on state machines (Statecharts) or Petri nets (DesignCPN, GrafCharts). Our interest in these notations is their ability to characterize the domain architecture, i.e., capture the structure and interfaces between different components of a control system, and how they can be integrated with capabilities focusing of implementation architectures. We will be examining the GrafChart toolset from Lund University under the full contract.

For the purpose of determining an appropriate platform for prototyping of the dependency tracking tool we have identified two candidates for evaluation: Meta-DOME, and G2. We will be investigating how Meta-DOME allows us to experiment with extensions to Meta-H to capture control domain specific semantic information. G2 is the implementation platform for the GrafChart toolset. G2 is a commercial product from GenSym in Boston and potentially provides a flexible environment to extend GrafCharts and define new modeling capabilities. Both these candidate will be evaluated as part of this subtask under the full contract.

Task 4: Avionic Applications Demonstrations

Objective

Task 4 will be primarily undertaken by Lockheed Martin Corporation. Nevertheless, CMU must undertake preliminary work to make the Lockheed Martin demonstrations of the INSERT technology feasible. Specifically, during the first three months, CMU will investigate the availability of appropriate Ada Compilers for use in the demonstration, and CMU will create requirements for the high performance but potentially faulty AMAS algorithm, and begin development of the Ada source code for the current AMAS algorithm.

summary of current progress to date

Upon investigation of the requirements of the extended contract, and in order to leverage previously performed work as much as possible, CMU has determined that Ada will not be used at this time in the implementation of the INSERT infrastructure. The non-embedded avionics demonstrations, where an external computer is interfaced with the avionics test-bench, will be performed in a hybrid language environment using the native tasking environment of the external computation system. Ada may be used to implement the AMAS control code, in order to maintain code level compatibility with the extended contract, year 3 end result.

In the event that Ada will be used in a mixed language environment, GNAT (GNU Ada-95) will be used to implement Ada language subroutines, which will be called from a non-Ada environment. The feasibility of this approach has been demonstrated previously at CMU.

Progress on creating an implementation of the AMAS algorithms has not been possible, due to contractual issues between CMU and Lockheed Martin Corporation (source material has been unavailable).

In order both to have an accessible test environment (to attempt to minimize travel to the Lockheed Martin facility), and to support the EDCS demonstration days, it was decided to develop a portable simulation / demonstration environment. The initial configuration of this environment will consist of a Sun Sparc-5 computer, running a F-16 Block 50 simulation environment developed by USAF Wright Labs. This simulation environment will contain the equations of motion, environment, and avionics simulations (including cockpit controls and displays).

Connected to this simulation will be a PC class machine to lend INSERT functionality to the system. This PC will communicate with the Sun F-16 simulation environment, and a commercial grade F-16 cockpit controls set (such as the Thrustmaster F-16 FLCS system, consisting of rudder pedals, stick and throttle components). Communications between the INSERT PC and the F-16 Sun simulation system shall logically replicate the interface with the Lockheed Martin Avionics Test Bench as much as is practical.

The basic demonstration environment will thus consist of 3 computers. The Sun F-16 simulation computer, the INSERT PC, and a Windows-95 PC that will interface to the cockpit controls subsystem and communicate with the INSERT PC.

The initial demonstration environment is planned to consist of a demonstration of this simulation environment. The INSERT PC is planned to perform 2 operations. It will pass cockpit control information to the Sun simulation environment. It will also implement a pilot relief autopilot system. This system will interface to the Sun simulation by modifying the cockpit control information. This is similar to the method used by the AFTI F-16 to implement the autopilot coupling.

Possible extensions to this simulation environment include it's integration with a Distributed Interactive Simulation "Stealth Viewer" system. This will extend the simulation environment and allow visualization of the simulation environment. This visualization can include terrain, as well as the relative locations of the F-16, weapons in flight, and any target of interest. This capability will be most useful when it comes to demonstrating and evaluating the AMAS weapon delivery algorithms, which will be implemented in Q3 and Q4 of 1997, as well as in outlying years.

Appendix 1: A Brief Summary of CMU's INSERT Program

Carnegie Mellon University, the Software Engineering Institute and Lockheed Martin Corporation will develop INSERT (Incremental Software Evolution for Real-Time Applications). The package will reduce the life-cycle development costs of complex, safety critical, real-time software applications by permitting safe on-line upgrades of new software, even though the new software may have residual errors. The INSERT capability package will facilitate a paradigm shift from today's approach of designing and testing for static requirements to a new approach of having a software framework that can be efficiently reconfigured to meet changing requirements and can facilitate the safe on-line insertion of new functionality. The INSERT package will be built on the Simplex Architecture (SA) (developed by the SEI) which exploits analytic redundancy to achieve fault tolerance, the Data Fusion Integrity Process (DFIP) (developed by Wright Laboratory), generalized rate monotonic scheduling, and other DARPA technologies. The SA exploits analytic redundancy to achieve fault tolerance. The DFIP technology has demonstrated the ability to tolerate a wide range of sensors faults and errors in processed sensor data.

The INSERT package will have the following integrated components:

1. Management of software design dependencies with computer-aided tracking of timing, resource, and semantic dependencies among components with respect to a given set of changes. Dependency analysis will evaluate the transitive closure of proposed software changes so that the worst-case impact will be known even before any software changes are attempted.

2. Distributed real-time middleware services to provide application-independent run-time and tool support for safely reconfiguring the software modules comprising an application. Software modules and their communications structure can be modified dynamically to respond to the changing needs of the application and changes in the environment, thus extending the concept of late binding to dynamic binding.

3. Dynamic real-time fault-management services which contain and manage faults so that at least a critical baseline of services is provided at any given point in time. Dynamic firewalls will isolate application modules from one another such that faults in changing software do not propagate. Active software safety nets will monitor performance and perform model-based data verification.

The INSERT package will be demonstrated yearly in the Lockheed flight simulator in the context of avionics applications. This class of applications is a single domain that contains many of the characteristics that contribute to high system evolution costs (e.g., real-time requirements, resource constraints, high assurance, high complexity and security).

- The Year 1 demonstration will provide implementations of the F-16 Automated Maneuver and Attack System (AMAS) in parallel with different hosts.

- The Year 2 demonstration will add DFIP support and GUI visualization tools into the middleware, and the switching logic will be extended to accommodate multiple computing modules.

- The Year 3 demonstration will provide a full-scale validation which will be used to validate the complete physical integration of INSERT technology into a production software environment with full implementation of dependency tracking. The INSERT middleware services will be hosted completely within a multiprocessing avionics computer.

Appendix 2: The Real-Time Push-Pull Communications Model

With the advent of networking technology, the demands for exchanging information over a distributed multiple processor environment are growing rapidly. Various types of information, such as voice, video control data, sensor data, real-time intelligence data, and text, are being transported widely across today's information and surveillance networks. Supporting these emerging applications require state-of-the-art distributed and multiple processor systems with common requirements such as timely processing, high availability, dynamic reconfigurability, scalability, global information access etc.

One of the most interesting applications in the distributed real-time information distribution and access domain is "Real-Time Push-Pull Communications." The objective of this effort is to provide a distributed infrastructure which enables a very flexible and powerful many-to-many communications model. Specifically, this will allow real-time processes on different machines to publish and receive information in real-time in a location-independent and protocol-transparent way. End-to-end timing guarantees can still be provided. This means that subscribers of particular message streams (identified by a "distribution tag"), need not know the location of the information producers (publishers) who are publishing on that distribution tag. Similarly, the information producers need not know who the information consumers (subscribers) are.

The Real-Time Publisher/Subscriber Communication Model

Throughout this project, the communication model will be based on the Real-Time Publisher/Subscriber Communications model for communication between information producers (publishers), information consumers (subscribers). A consumer can consume information from multiple sources, while a producer can produce information for multiple consumers. A producer can also be a consumer and vice-versa. A data channel represented by "a distribution tag" represents each category of data that is available in the system. A publisher publishes on a distribution tag, and subscribers to that tag can consume the information published on that tag. However, a publisher need not know who the subscribers to its published information are, and a subscriber need not know who the publishers of its consumed information are. In other words, the publisher-subscriber communication model allows a general many-to-many communications model.

In the real-time publisher-subscriber communication model, timing delays for communications between an information producer and an information consumer are predictable, analyzable *and* efficient. As may be expected, this analyzability is based on the assumption that the communication demands are pre-defined or can be known using schemes such as admission control.

The implementation of the real-time publisher-subscriber communication model allows the information about publishers and subscribers to be stored as close to the publisher as possible (namely its own address space on its own node). This yields significant performance benefits. A distributed fault-tolerant name-service hidden from the application interface allows the physical communications among publishers and subscribers to occur. This name-service allows processor nodes to fail and/or to (re)join the system. When failures happen or when nodes (re)join, the naming service continues to function and real-time publication/subscription on distribution tags continue to function normally (except, of course, on the nodes that failed or are trying to join).

The Real-Time Push-Pull Communications Model

We are now working on a broader generalization of the real-time publisher/subscriber communication model, namely the **Real-Time Push-Pull Communications** Model.

The real-time publisher/subscriber communication model can be considered to represent "push communications" where data is "pushed" by publishers. As a result, subscribers can obtain information only at the rate at which data is being pushed. This is the current implementation of the INSERT software architecture where the I/O sources, feedback controller modules and the semantic checkers all must operate at the same frequency[1]. This can be very limiting in many cases where different clients have different processing power and/or widely varying communication bandwidth (because of connectivity to a low bandwidth network like a telephone modem or an encrypted satellite link). If consumers did not have the same power, a publisher must either falsely assume that they all have the same capability or publish two (or more) streams to satisfy consumers with different capabilities.

It would be very desirable if a client with a relatively low processing power and/or communication bandwidth is able to consume published data at its own preferred rate. In other words, the data reaching this client depends on its own needs, and not that of the publishing volume/rate of the publisher. Also, the real-time publisher/subscriber model is very *synchronous*: subscribers normally block on a (distribution tag) port waiting for data to arrive, and publishers produce data at the rates that they determine, and the published data is immediately sent to the subscribers on that distribution tag.

The Push-Pull Communications model addresses both of the above concerns. It allows consumers on the same data streams to receive *and* process data at independent (locally determined) rates. As a result, client with high or low processing power and/or high or low network bandwidth can consume data on a stream. In addition, this can happen without the knowledge of the data producers who do not have to distinguish between the capabilities of the receiving consumers. In the push-pull communication model, data can be either "pushed" by an information producer or "pulled" by an information consumer. A "pulling" consumer can choose to consume data at a rate lower than the data production rate. In the extreme, a pulling consumer can choose to only consume data *asynchronously.*

The Real-Time Push-Pull Communications model adds predictability, analyzability and timing guarantees to the push-pull communication model. This is accomplished by specifying the task and communications parameters to be used, the use of real-time scheduling schemes (priorities and synchronization protocols which bound and minimize priority inversion).

In addition, the real-time push-pull communications model adds the following interfaces.

Asynchronous Pull-Message

This call is used by a consumer to "pull" the latest message published by a particular publisher. This call will also allow

- The consumer can specify the number of messages that should be pulled. By default, the

[1]A subscriber may choose to operate at a different lower frequency by, for example, skipping every other published datum on a subscribed tag. However, for this to happen, the subscriber must still receive and "consume" the datum albeit in a trivial "drop-it" fashion.

number of messages is 1, but more than 1 message can be pulled on demand.

- If there are multiple publishers on the distribution tag, the consumer can specify a particular producer from which the message must be pulled.

Synchronous Pull-Message
This call is used by a consumer to "pull" the latest message published by a particular publisher *automatically* at periodic intervals. Even if the publisher is publishing at a different (lower or higher) rate, the middleware service will ensure that the latest messages on the tag published at the time of needed consumption are delivered. This call will allow the following: will be consumed.

- The consumer can specify the period at which the messages should be received.
- The consumer can specify the number of messages that should be pulled. By default, the number of messages is 1, but more than 1 message can be pulled on demand.
- If there are multiple publishers on the distribution tag, the consumer can specify a particular producer from which the message must be pulled.

Store-Message
This call can be used explicitly by publishers to "store" a message to be "pulled" later by a consumer. The underlying middleware infrastructure will store only a finite number of messages based on the underlying implementation and available system resources.

Attributes of a Pull Tag
Each distribution tag now also has "Pull attributes" associated with it. These attributes can be modified or queried by a publisher or a subscriber.

Publisher-Modify-Pull-Attribute
This call allows publishers to specify the "pull capabilities" on a distribution tag that must be supported by the push-pull communications middleware services. For example, a publisher can specify the maximum number of messages that can be stored for pulling by a consumer. The publisher can also specify whether each message should be time-stamped and/or have a sequence number.

Subscriber-Modify-Pull-Attribute
This call allows subscribers to specify the "pull capabilities" on a distribution tag that it expects to "pull" information from. For example, a subscriber can specify the maximum number of messages that need to be stored for its pulling. It must be noted that this maximum number of messages that will be pulled by a subscriber can *exceed* the maximum number of messages that were specified by the publisher on that distribution tag. The implications of this feature are not always intuitive and will be discussed next.

Query-Tag-Attribute
This call allows a publisher or subscriber to query the "pull" attributes specified on a distribution tag by a publisher or subscriber.

Proxy Publishers
A new entity called a "Proxy Publisher" will be created in the real-time push-pull communication

model to cater to the special needs of "pull communications". The "proxy" acts as an intermediary between "push" and "pull": it acts as a consumer to the publishers, and as a publisher to the pull client. As a result, the publisher and pull client can actually have very different needs: processing power, bandwidth requirements, message buffer size and rates of production and consumption.

As stated in the previous section, the consumer can actually "pull" more messages than are actually requested for storage by a publisher. This is because the proxy publisher for the pull client can store messages that are published by the producer and store the necessary number of messages to satisfy the needs of the pull client.

Concluding Remarks

We have generalized the real-time publisher-subscriber communications model to present a new many-to-many communications model called the "real-time push-pull communications model." The real-time publisher-subscriber communications model can be considered to be "push technology" where information producers are the initiators of message transmission, and information consumers synchronize and must consume the information produced. "Pull communications" enables the power of the "opposite direction" where the information consumers can specify and consume only those messages that they want and only at the rates that they can consume (independent of the rate at which the information is produced). This allows information sources and sinks with a wide range of processing power, network bandwidth and synchronicity needs to co-exist within a single system. The real-time aspects of the model involve the use of real-time scheduling and resource management principles, appropriate priority assignment, specification and management of end-to-end timing constraints, a priori scheduling analysis. As a result, a distributed system with very general and powerful many-to-many real-time communications can be built such that the real-time behavior of the system can be predictable, analyzable and guaranteed in advance.

Status

The design of the model is complete and implementation is in progress.